Performance analysis of compaction techniques for map representation in geographic databases

P. Boursier, M. Scholl

INRIA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P 105
78153 Le Chesnay Cedex
France
Tél. 954 90 20

Rapports de Recherche

N° 58

PERFORMANCE ANALYSIS OF COMPACTION TECHNIQUES FOR MAP REPRESENTATION IN GEOGRAPHIC DATABASES

Patrice BOURSIER
Michel SCHOLL

Mars 1981

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. 954 90 20

Rapports de Recherche

N° 58

PERFORMANCE ANALYSIS OF COMPACTION TECHNIQUES FOR MAP REPRESENTATION IN GEOGRAPHIC DATABASES

Patrice BOURSIER
Michel SCHOLL

Mars 1981

PERFORMANCE ANALYSIS OF COMPACTION TECHNIQUES FOR MAP REPRESENTATION IN GEOGRAPHIC DATABASES

=:=:=:=:=:=:=

Patrice BOURSIER*, Michel SCHOLL**

ABSTRACT :

A new compaction scheme for representing cellular thematic maps in geographic databases is presented. The memory saving provided by this method, called Hierarchical Compaction Scheme (HCS) is analyzed and compared to that of two more classical cellular schemes. HCS not only provides a significant compaction ratio but also renders processing operations such as map intersection or union easier. It also allows the use of variable scale.

RESUME :

Une nouvelle méthode de représentation compacte des cartes géographiques en mémoire est présentée. Le gain en espace mémoire que procure cette méthode cellulaire appelée Compactage Hiérarchique (CH) est analysé et comparé au gain obtenu avec deux méthodes cellulaires plus classiques.Non seulement CH procure un gain en espace mémoire important, mais encore il rend les opérations ensemblistes sur les cartes, telles que l'intersection ou l'union plus faciles. Il permet également l'utilisation d'échelles variables.

* ORSTOM, 93140 Bondy, France

**INRIA, 78153 Le Chesnay Cedex, France

1 - INTRODUCTION

The existing techniques for representing and processing map data in geographic databases may be partitionned into two categories as defined in [13] :

- in the polygonal techniques [6,9,11] any object in the map (corn field or town, for example) is represented by its contour. Coding schemes are used for storing the contour lines.

- on the contrary, in the cellular techniques [1,3] a gridding is first applied, then all the cells "belonging" to the object are stored. The scheme compactness becomes therefore of paramount importance, especially if the purpose of encoding is primarily storage or transmission of data. However, the storage efficiency resulting from a good compaction scheme may be outweighted by an increased complexity in processing, encoding and decoding the data.

The techniques studied in this paper fall into the second group. A thematic map is used for representing a geographic attribute (such as land use or slope) that may be divided into elementary values or classes of values to each of which corresponds an elementary map [3]. Each of these elementary maps is coded by means of a binary matrix that may be processed as a binary string. There are then as many binary strings as there are attribute values and it is necessary to reduce this considerable amount of storage that would otherwise prohibit the use of such methods.

The Hierarchical Compaction Scheme (HCS), introduced in [4], fulfills this compactness objective. This technique, derived from a well-known binary string compaction technique [8], allows the database designer to represent a map under various scales. Besides, the set operations on maps are easily processed with such a data structure (see Section 2).

HCS is compared with respect to storage efficiency to two-more classical compaction techniques. The first one is a variant of Run-Length Coding (RLC) [10] while the second one called Mixed Compaction Scheme (MCS) has been implemented in [3]. An analysis of compaction ratio under HCS, RLC and MCS is reported in [4] which is based on very restrictive assumptions. We give below a new analysis of compaction ratio under RCL (Section 3) and MCS (Section 4) based on less restrictive assumptions (HCS could not be analyzed under the latter assumptions).

In order to test the validity of the analysis, memory saving under these three techniques has been estimated from a sample of data actually stored in a geographical information system built for the Emilia-Romagna region [2,3] (Section 5).

The analytical prediction being very pessimistic, a simulation was run which leads to a more realistic prediction of the compaction ratio (Section 6).

In the sequel of the paper, we assume the data to be compacted is a N bits long string and the expected memory saving (compaction ratio) G(N) is given by : $G(N) = 1 - S(N)/N$, where S(N) is the expected length of the string after compaction.

2 - HIERARCHICAL COMPACTION SCHEME (HCS)

We describe below two variants of this scheme respectively denoted by HC_0 and HC_1.

2.1 - Description

The coding scheme is derived from the following block-compaction technique [7,8] : the initial string C_0 is cut into blocks of equal length u bits. Without loss of generality, assume $N = u^{\ell}$. After compaction, C_0 is represented by a string C_1 composed of a "data string" J_1 of smaller size preceded by an index I_1. The index is organized as follows : there is a bijection between the $\frac{N}{u}$ blocks of the initial string C_0 and the $\frac{N}{u}$ bits of the index. Each index bit is set to zero if the corresponding block in C_0 is full of zeros. Such a block is called a 0-block. Otherwise, the index bit is set to one. The data string J_1 is obtained by suppressing in the initial string all the 0-blocks. Figure 1a and 1b display an example of string and its representation after compaction.

We generalize this scheme by compacting in turn the index I_1 using the same above technique. I_1 is compacted into an index I_2 and a data string J_2. Therefore, after two compactions, the initial string C_0 has been transformed into a string C_2 composed of an index I_2, followed by a data string J_2, followed by the data string J_1. I_2 may be in turn compacted, etc...

Figure 1c shows the representation of string C_0 after two compactions.

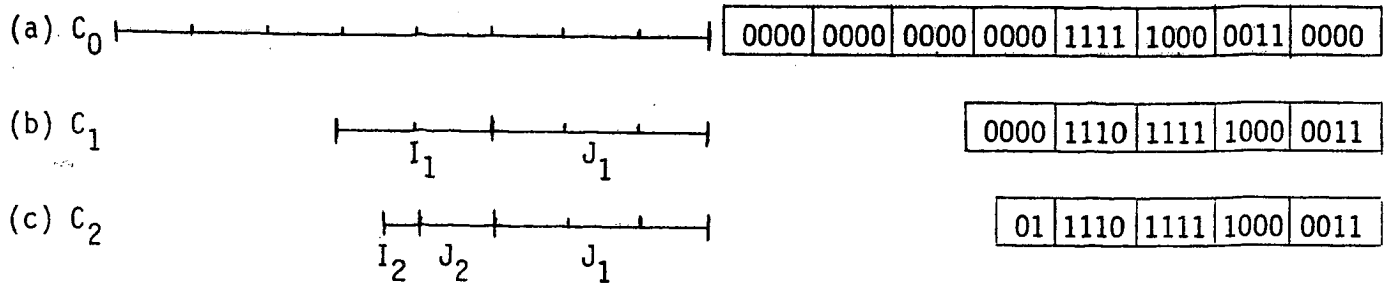

Figure 1 : HCS, variant HC0, u = 4

As a matter of fact, for a given string of size N, there is an optimal number of compactions, n_0 ($n_0 \leq \ell$), which maximizes the memory saving G(N). On the example of Figure 1, $n_0 = 2$.

The memory saving achieved by this method is obviously better when the proportion of "0-bits" in the string, p, is high (greater or equal to 90 per cent). Conversely, when p is very low, the 1-bits have to be compacted in the place of the 0-bits.

In constrast with the above method called HC0, in the variant called HC1, both 0-blocks and 1-blocks are compacted. At each compaction level, a "data-block" (of size u bits) is represented in the index by 2 bits, as follows : the codes 00, 11 and 01 are respectively assigned to a 0-block, a 1-block and any other block.

Figure 2 illustrates variant HC1 on the example of Figure 1.

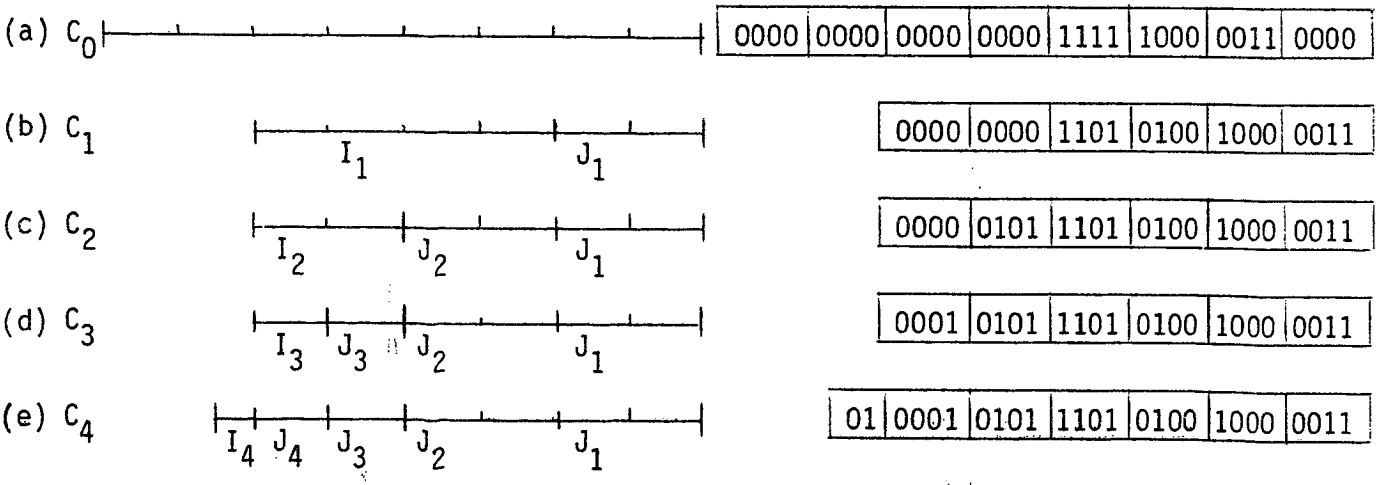

Figure 2 : HCS, variant HC1, u = 4

The optimal compaction level is now $n_0 = 1$ with a size of 24 bits for C_1. Another variant of HCS may be found in [5].

2.2 - Analysis

We give in this section, the expected memory saving G(N) as defined in Eq. (1) under the following assumption :

A1 : Let X_i be a random variable representing the value of the ith bit in the string. We assume the X_i's, $i \in [1,N]$ are independent and identically distributed : $P\{X_i = 0\} \triangleq p$.

Consequently, the probability for any block to be a 0-block (resp. 1-block) is equal to p^u (resp. $(1-p)^u$).

The validity of this assumption will be tested on a sample of actual data in Section 5.

It is shown in [4] that the memory saving achieved by HC0 is equal to :

$$(2) \qquad G(N) = \sum_{i=1}^{n_0} \frac{up^{u^i} - 1}{u^i}$$

where n_0 is such that :

$$(3) \qquad up^{u^{n_0}} > 1 \text{ and } up^{u^{n_0+1}} \leq 1$$

while for HC1, we have :

$$(4) \qquad G(N) = \sum_{i=1}^{n_0} \frac{u/2\left[p^{u^i/2^{i-1}} + (1-p)^{u^i/2^{i-1}}\right] - 1}{u^i/2^i}$$

where n_0 is such that :

$$(5) \qquad \begin{cases} u/2\left[p^{u^{n_0}/2^{n_0-1}} + (1-p)^{u^{n_0}/2^{n_0-1}}\right] > 1 \\ u/2\left[p^{u^{n_0+1}/2^{n_0}} + (1-p)^{u^{n_0+1}/2^{n_0}}\right] \leq 1 \end{cases}$$

2.3 - Discussion

G(N) is plotted versus u (Eqs (2) and (4)) for both HC0 (plain curves) and HC1 (dotted curves), and for various values of p in Figure 3. The curves are labelled by the corresponding values of n_0.

First, note that a significant memory saving is obtained with these compaction schemes. Indeed, if p = .999, about 99 per cent of the memory is saved and if p = .99, we need only $\frac{N}{10}$ bits for storing a string of N bits (G(N) $\simeq$ .90 with HC0). However G(N) decreases fast as p decreases. Intuitively, even for values of p around .95, one should expect a saving larger than that predicted by our model. In Section 5, we show that indeed the model is pessimistic when p is not close to 1.

Second, observe that the larger u, the smaller the memory saving. Then, on the one hand, u should not be too large (the larger u, the smaller the probability of a block being full of zeros, then the lower G(N)). But on the other hand, if u is too small, the saving may be outweighted by the index overhead (increasing as u decreases). Thus an optimal value of u exists which maximizes G(N).

Third, it is worth nothing that HC0 always provides a better performance than HC1 does, i.e. the saving due to the 1-blocks compaction does not outweigh the index extraoverhead with HC1 (two bits are needed for coding the index with HC1 while only one bit is necessary with HC0).

3 - RUN-LENGTH CODING (RLC)

This compaction technique is derived from the well-known "run-length coding" technique described in [10], when only two values may appear in the initial string, 0 and 1.

3.1 - Description

Any string to be compacted being a sequence starting with a run of zeros (ones) followed by a run of ones (zeros) in turn followed by a run of zeros (ones), etc..., the RLC technique consists in representing the sequence of runs by their respective lengths, each length being coded with one block of u bits.

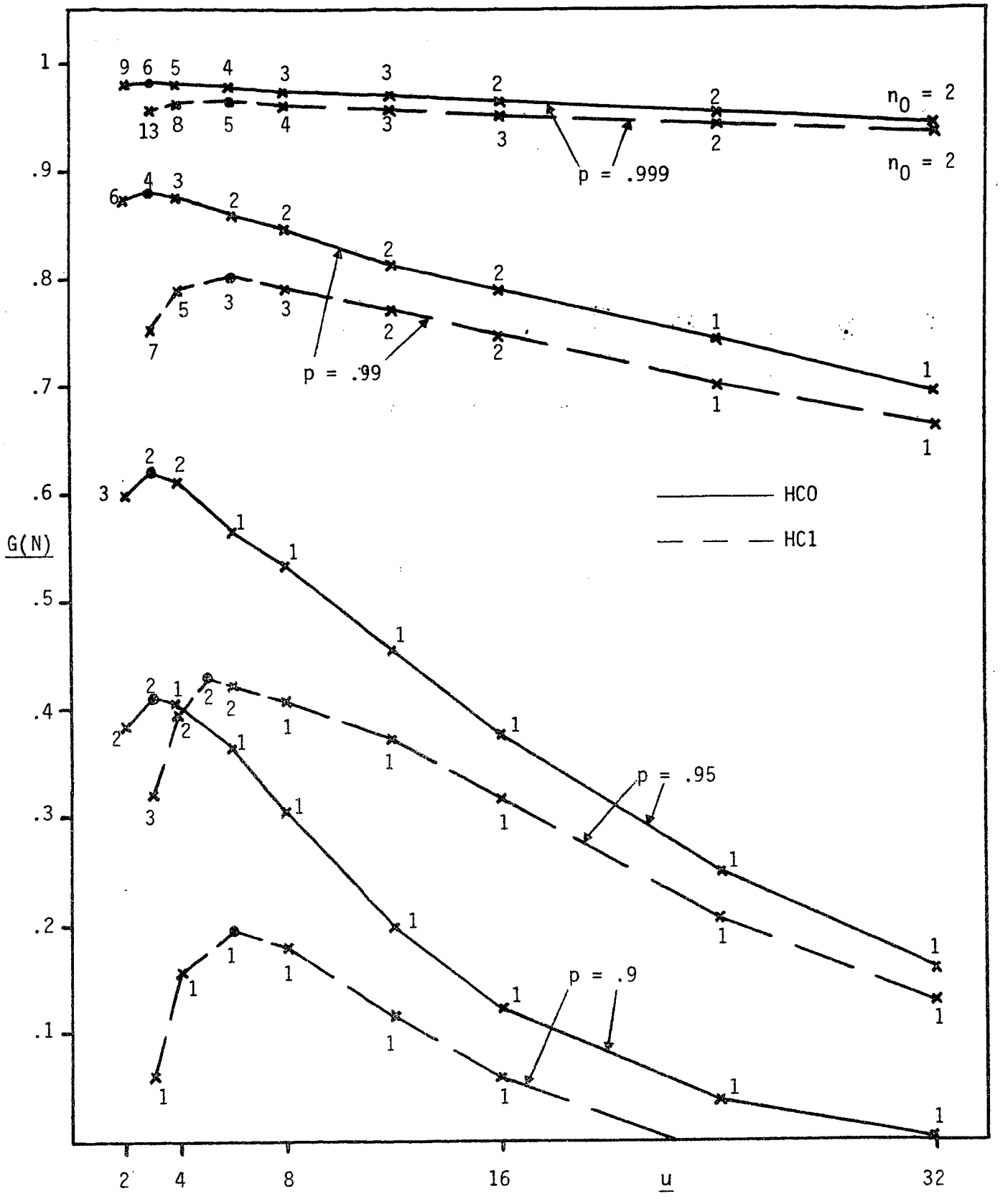

Figure 3 : HCS : memory saving versus block size u for various values of p

In the case where the length is larger than (2^u-1) bits, more than one block is necessary. More generally, if the length is such that $i(2^u-1) < \ell \leq (i+1)(2^u-1)$, then we need (i+1) blocks where the i first blocks are full of 1. Figure 4 shows examples of run-lengths with their respective representations.

Run-Length (bits)	Encoding
1	0000
5	0100
15	1110
16	1111 0000

Figure 4 : Run-Length encoding, u = 4

Figure 5 shows how the string C_0 of Figure 1a is represented after a run-length encoding has been applied (u = 4). Note an additional bit is necessary to indicate whether the string starts with a 0-run or a 1-run.

C_4 : 0 1111 0000 0100 0100 0001 0011

Figure 5 : Run-length coding, u = 4

3.2 - Analysis

Assume without loss of generality the initial string C_0 starts with a 0-run. C_0 may be considered as a sequence of subchains, each of them including a 0-run followed by a 1-run.

Let us also assume :

A2 : the lengths of the subchains are independent and identically distributed according to any distribution function.

The subchains sequence may then be considered as a reward renewal process [14] where the occurrence of the first 0-bit of a subchain is a renewal point and the associated reward is the representation "cost" of this subchain in number of blocks of size u bits. We therefore have :

$$(6) \qquad \lim_{N\to\infty} \frac{S(N)}{N} = \frac{E(Y)}{E(X)}$$

where E(X) is the average length of a (non-compacted) subchain and E(Y) the average "cost" of the subchain compacted representation, i.e. the average number of bits necessary for the compacted representation.

Eq. (6) is true for any length distribution (as long as assumption A2 holds). In appendix A, E(X) and E(Y) are derived for a particular case corresponding to the following assumption (less restrictive than assumption A1 of Section 2) :

Recall X_i is a random variable representing the value of the ith bit in the string. Then assume X_i is a Markov chain of order 1, i.e. more specifically, we have :

$$\text{A3} \quad \begin{cases} P\{X_{i+1} = 0 \mid X_i = 0\} \triangleq p_0 \\ P\{X_{i+1} = 1 \mid X_i = 1\} \triangleq q_1 \end{cases}$$

Then the expected length of a 0-run is $p\bar{\ell} = 1/1-p_0$, while that of a 1-run is $(1-p)\bar{\ell} = 1/1-q_1$, where p is the "proportion" of 0-bits in the string and $\bar{\ell}$ is the expected length of a subchain.

Then we have for N large (see Appendix A) :

$$(7) \quad E(X) = (2-p_0-q_1) \,/\, [(1-p_0)\,(1-q_1)]$$

$$(8) \quad E(Y) = u \sum_{j=0}^{j_{max}-1} \left[p_0^{\,j(2^u-1)} + q_1^{\,j(2^u-1)} \right], \text{ where } j_{max} = \lceil N/(2^u-1) \rceil$$

Combining Eqs (6), (7) and (8), we have :

$$(9) \quad \lim_{N\to\infty} \frac{S(N)}{N} = \frac{(1-p_0)\,(1-q_1)}{2-p_0-q_1}\, u \sum_{j=0}^{j_{max}-1} \left[p_0^{j(2^u-1)} + q_1^{j(2^u-1)} \right]$$

The term in the sum rapidly becomes negligible as j increases. Consequently, the right-hand side of Eq. (9) is a good approximation of S(N)/N as N is large.

The memory saving is then :

$$(10) \quad G(N) \simeq 1 - \frac{(1-p_0)\,(1-q_1)}{2-p_0-q_1}\, u \sum_{j=0}^{j_{max}-1} \left[p_0^{j(2^u-1)} + q_1^{j(2^u-1)} \right]$$

3.3 - Discussion

G(N) is plotted (Eq. (10)) versus u in Figure 6 for $\bar{\ell}$ = 100 and for various values of p. As for HCS, there is an optimal block size : indeed, if u is chosen too small, many blocks may be necessary to code a run-length. If u is chosen too large, space is lost into the blocks for coding lengths which, most of the time, are much smaller than (2^u-1).

Note that the memory saving is more sensitive to variations in the block size u than with HCS. In Figure 7, G(N) is plotted versus (1-p) for u = 8 and for various values of $\bar{\ell}$. Observe that G(N) is not sensitive to a variation in p. Indeed, a subchain is always represented by two blocks (one for the 0-run and one for the 1-run), although very seldom we need more than 1 block for representing a 0-run, (when p is very close to 1). As expected, G(N) increases as $\bar{\ell}$ increases : obviously, since a subchain is always represented by two blocks, the larger the subchain, the larger G(N).

4 - MIXED COMPACTION SCHEME (MCS)

4.1 - Description

The scheme studied in this section is very similar to that used in the geographic database described in [3]. The initial string C_0 is cut into blocks of size u bits.A block is either a 0-block (full of zeros), or a 1-block (full of ones), or a non-monotonic block. C_0 may then be viewed as a sequence in any order of 0-runs (made of one or more successive 0-blocks), 1-runs (made of one or more 1-blocks), and non-monotonic runs (made of one or more non-monotonic blocks).

A 0-run (1-run) is represented by one word of size (u+2) bits where the two first bits are equal to 00 (11) and the last u bits represent the run length in number of blocks. Non-monotonic runs are represented by as many words of size (u+2) as there are blocks in the run, where the two first bits are equal to 01, while the non-monotonic block (u bits) is copied into the last u bits of the word. If the number of blocks in a 0-run (1-run) is greater than 2^u, then two or more successive words are necessary as in the RLC technique (see Section 3.1). Figure 8 illustrates the MCS technique on an example where u = 2.

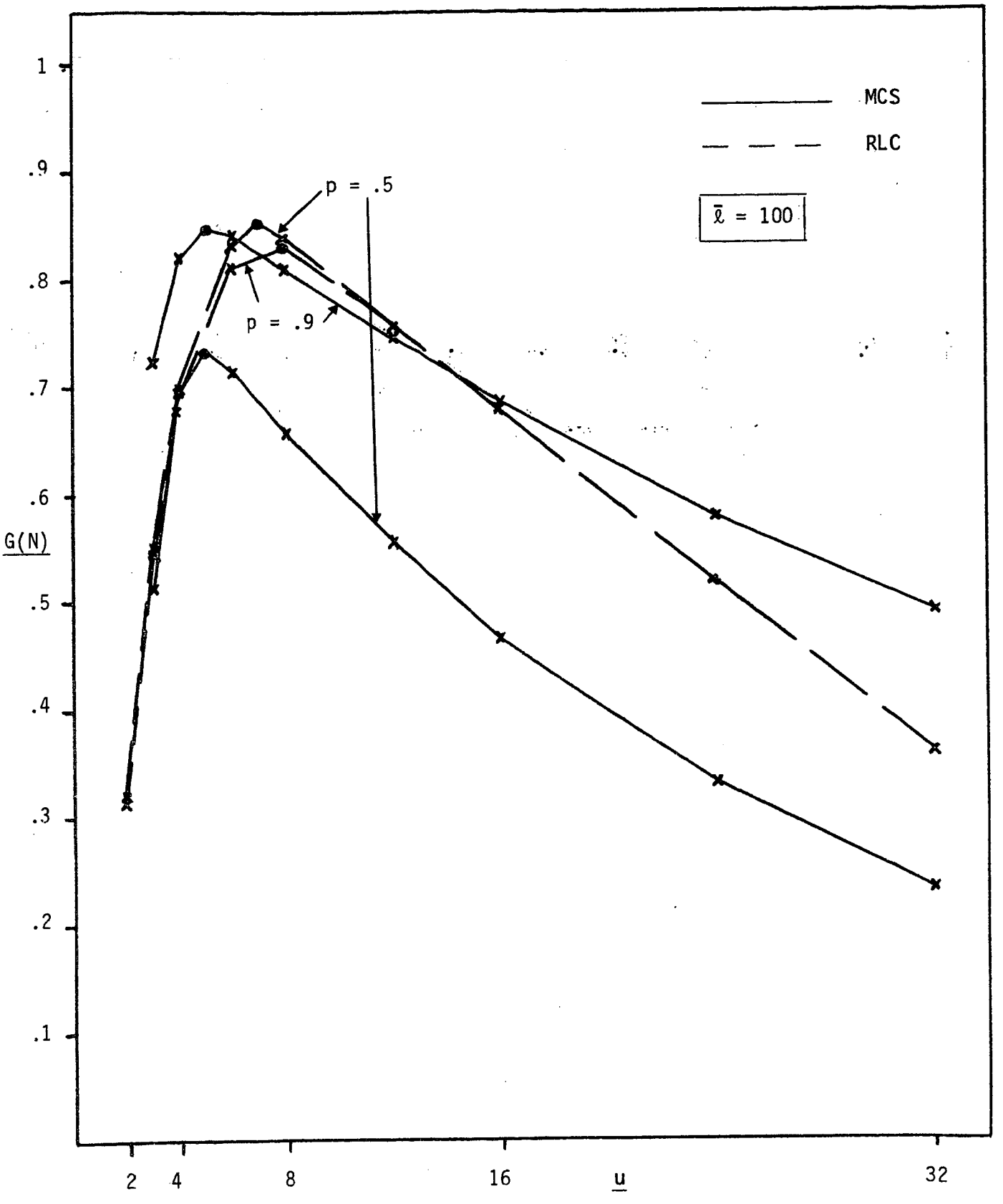

Figure 6 : RLC and MCS : memory saving versus block size for $\bar{\ell}$ = 100 and for various values of p

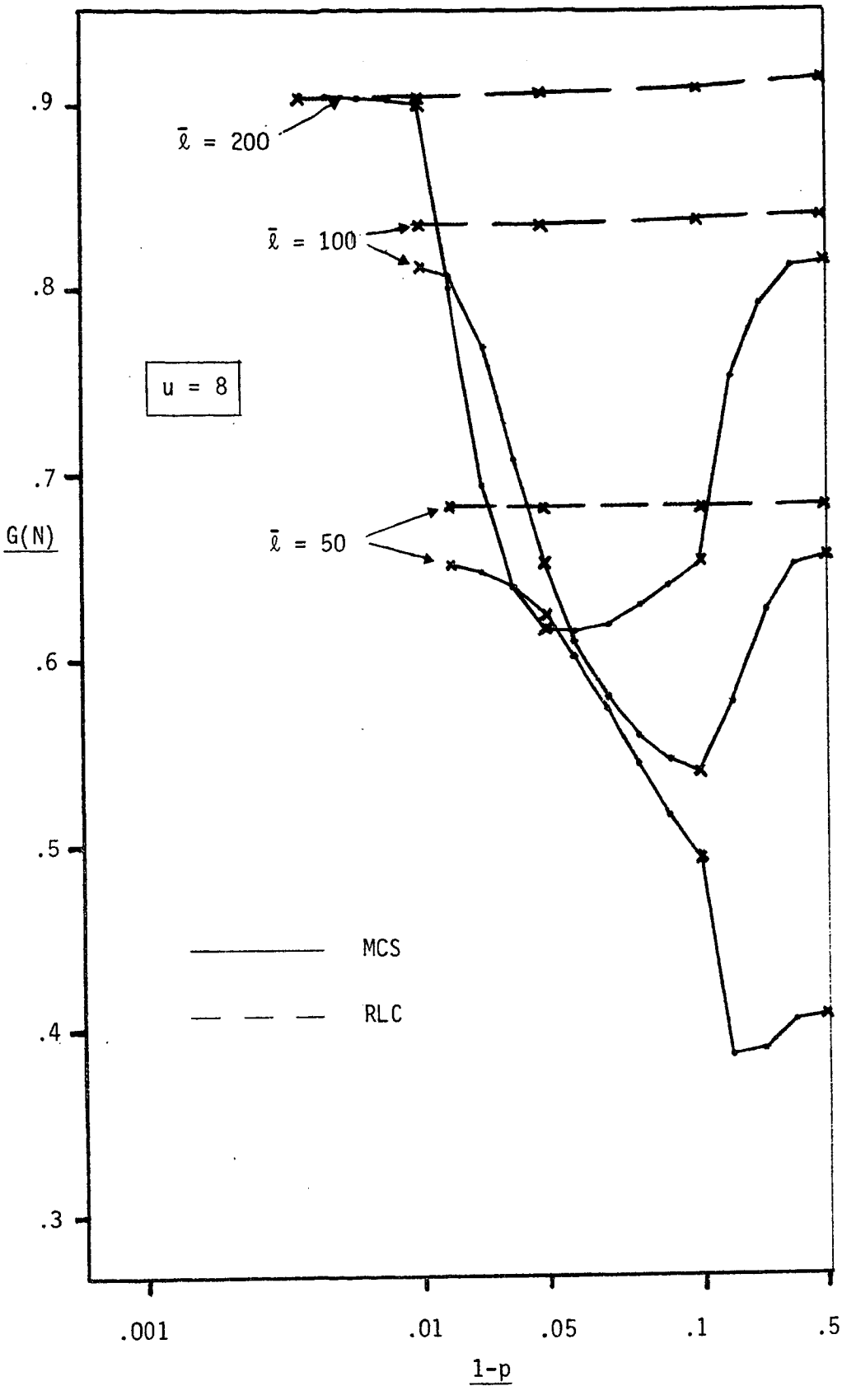

Figure 7 : RLC and MCS : memory saving versus average proportion of 1-bits (1-p) for u = 8

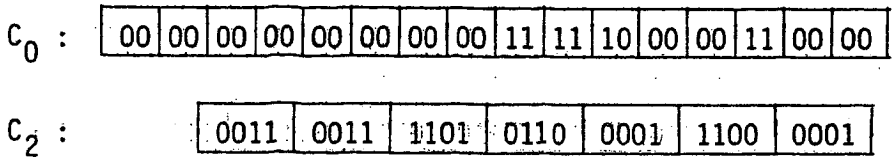

Figure 8 : MCS technique, u = 2

Finally, note that this technique combines block compaction and run-length encoding.

4.2 - Analysis

If assumption A3 (see Section 3.2) is verified, then the sequence of blocks in string C_0 may be viewed as a Markov Chain with four states α, β, γ, δ respectively corresponding to the occurrence in the string of a 0-block, a 1-block, a 10-block (non-monotonic block ending with a 0-bit) and a 01-block (non-monotonic block ending with a 1-bit).

The transition probabilities are given below as a function of p_0, q_1 (see Section 3.2) and of the following probabilities :

$$a \triangleq P\{X_{i+u} = 0 \mid X_i = 0\} \text{ and } b \triangleq P\{X_{i+u} = 1 \mid X_i = 1\}$$

a (b) is the probability a block ends with a 0-bit (1-bit) given the previous block ended with a 0-bit (1-bit).

We have :

$$P_{\alpha\alpha} = P_{\gamma\alpha} = p_0^u$$
$$P_{\alpha\beta} = P_{\gamma\beta} = (1-p_0)q_1^u$$
$$P_{\alpha\gamma} = P_{\gamma\gamma} = a - p_0^u$$
$$P_{\alpha\delta} = P_{\gamma\delta} = 1 - a - (1-p_0)q_1^{u-1}$$
$$P_{\beta\alpha} = P_{\delta\alpha} = (1-q_1)p_0^u$$
$$P_{\beta\beta} = P_{\delta\beta} = q_1^u$$
$$P_{\beta\gamma} = P_{\delta\gamma} = 1 - b - (1-q_1)p_0^{u-1}$$
$$P_{\beta\delta} = P_{\delta\delta} = b - q_1^u$$

One easily shows [5] that :

$$a = \frac{(p_0 + q_1 - 1)^u (1 - p_0) + (1 - q_1)}{2 - p_0 - q_1}$$

$$b = \frac{(p_0 + q_1 - 1)^u (1 - q_1) + (1 - p_0)}{2 - p_0 - q_1}$$

Observe that, if $p_0 = 1-q_1 = p$, we have, as expected, $a = 1-b = p$ (the probability $X_i = 0$ for all i, is equal to p ; we are back to assumption A1 of Section 2).

The resulting memory saving is given by (see Appendix B) :

$$(11) \quad G(N) = 1 - \frac{u+2}{u}\left[\frac{p_0^u(1-p_0^u)}{1-p_0^{2^u u}} + \frac{q_1^u(1-q_1^u)}{1-q_1^{2^u u}} + (a-p_0^u) + (b-q_1^u)\right]$$

4.3 - Discussion

Basically, as u increases, G(N) has the same behaviour with MCS as with RLC (see Figure 6) and with HCS (see Figure 3). Nevertheless G(N) does not drop as much with MCS as with RLC, when u increases. As for RLC, the larger the subchain length is, the larger G(N) is (see Figure 7). Note however, G(N) is very sensitive to the proportion p of 0-bits : a minimal value of the compaction ratio is observed for a value of p increasing with the average subchain length ($\bar{\ell}$, see Figure 7). Indeed, as p decreases, (i) space is lost into the words for coding number of monotonic blocks which most of the time are much smaller than 2^u-1, (ii) if p is not too close to .5, the saving due to the 1-bits compaction does not outweigh the coding overhead (equal to 2 bits).

5 - EXPERIMENTAL VALIDATION OF THE ANALYSIS

The "white noise" assumption of Section 2 (Assumption A1) is obviously pessimistic since it supposes no correlation exists between the values of two successive bits in the string. Assumption A3 (see Section 3) is less restrictive and should lead to better predictions. In order to check how much pessimistic the theoretical performance is and to test whether the qualitative analysis conclusions are verified, the schemes performance has been estimated using data issued from a sample of 38 elementary maps actually stored in the geographic database described in [3]. This system has been designed as an aid

for land-use planning of the Italian Emilia-Romagna region [2]. Each map is represented by a binary string of about N = 180,000 bits which is compacted according to the three techniques.

For the Hierarchical Compaction Scheme (HCO), the performance is computed by choosing for each map the optimal level of compaction n_0. Note that, in practice, one may prefer to choose the same level of compaction for all maps. Then the expected memory saving would be lower.

The "white noise" assumption implies the 0-runs (resp. 1-runs) are geometrically distributed with mean 1/(1-p) (resp. 1/p). This assumption is far from being justified. Indeed, the average length of a 0-run (taken over all maps) is found to be equal to 78.03 while a geometrical distribution would lead to a mean length equal to 22.73 (p = .956 ; for more details, see [5]).

The theoretical performance (from Eqs (2), (10) and (11)) and the actual performance (from measurement) are compared on Figure 9 where G(N) is plotted versus u for HCO, RLC and MCS.

From Figure 9 , one may draw the following conclusions :

(i) Quantitatively speaking, Assumption A1 is very pessimistic : for HCS, the measured gain in memory is by far superior to that predicted by the analysis of Section 2. Assumption A3 allows a good performance prediction for RLC, but is still pessimistic as far as MCS is concerned. However, it allows a much better prediction of MCS performance than Assumption 1 does (see [4]).
Observe HCO and MCS provide a higher compaction ratio than RLC does.

(ii) Qualitatively speaking, an optimal block size exists for the three techniques and only RLC performance is very sensitive to the block size. This was predicted by the analysis.

More results are given in [5]. In particular it is shown, for this sample of data, that 95% of the optimal memory saving is obtained if we choose the compaction level n_0 to be equal to 2, with HCS.

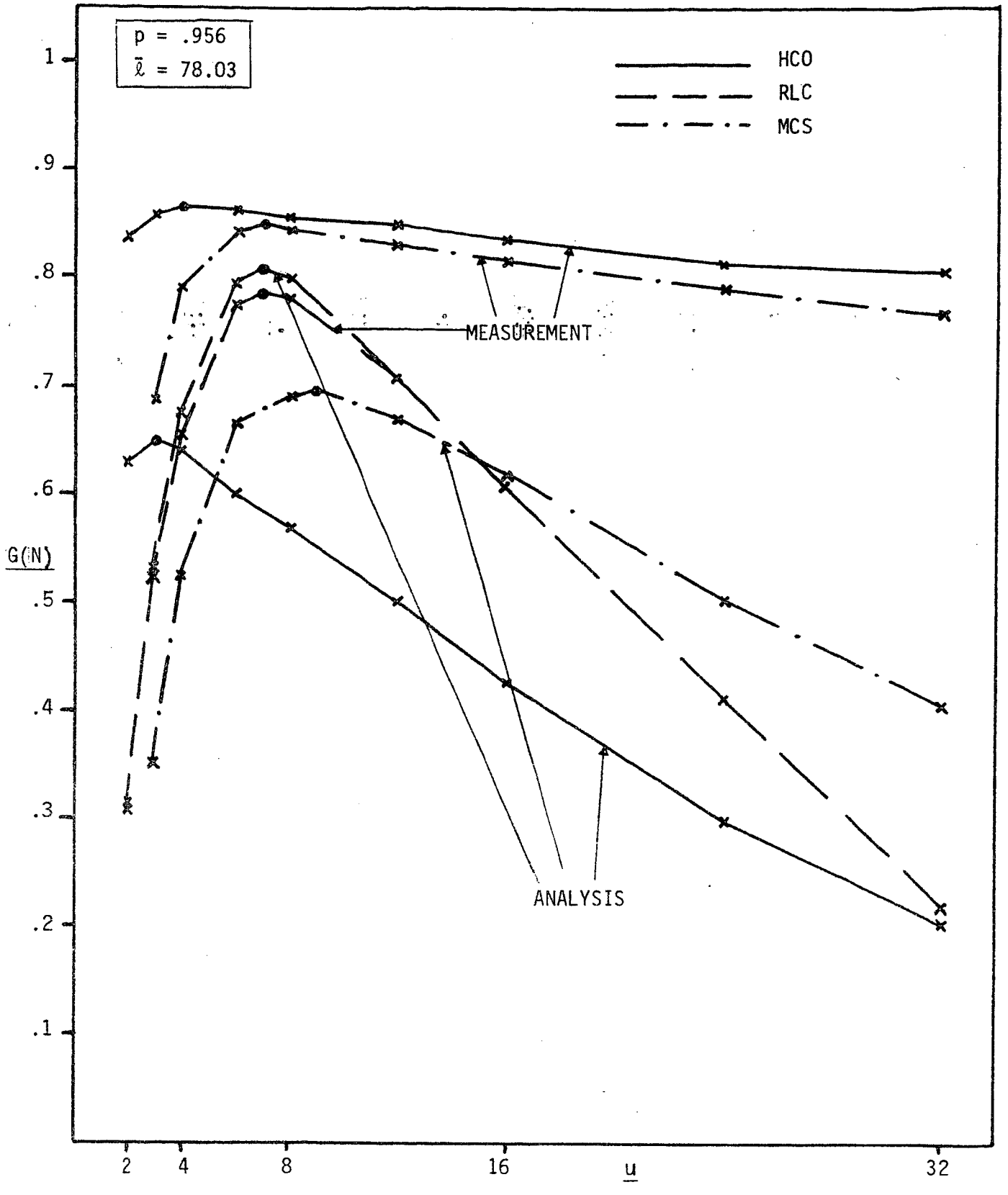

Figure 9 : HCO, RLC, MCS : memory saving versus block size. Comparison between Analysis and Measurement

6 - SIMULATION AND FURTHER COMPARATIVE STUDY

The influence of the run lengths on the compaction ratio provided by the three techniques has been studied in [5] by means of a simulation experiment. Runs were generated according to an hyperexponential distribution. No correlation was assumed between any two successive runs. The performance predicted by this simulation is in agreement with that reported in Section 5 and obtained from an actual sample of data, since the error on the optimal memory saving is less than 7% [5].

Measurement showed the superiority of HC0 and MCS in terms of memory saving, when the proportion of 0-bits is high ($p = .956$, see Figure 9). However, when p decreases, RLC and MCS are expected to ouperform HC0, a variant of HCS in which only 0-bits are compacted. HC1, the other HCS variant in which both 0-bits and 1-bits are compacted should then provide a much better performance than HC0 does, when p is as small as .5.

This is confirmed by the simulation study : in Figure 10, G(N) is plotted versus (1-p) as obtained from simulation for the four techniques, namely HC0, HC1, RLC and MCS. A block size of $u = 8$ bits has been chosen for all schemes.

From Figure 10, clearly as p decreases beyond .9, HC0 performance drops drastically, while HC1 outperforms HC0 (G(N) stays larger than .7, for $\bar{\ell} = 100$).

Observe, the simulation is in agreement with the analysis as far as RLC and MCS are concerned (see Figure 7). In particular, with RLC, the memory saving increases only slightly as p decreases (when p decreases, the probability it takes more than one block to code a 0-run is more and more negligible : this event happens when a run has a length greater than 255 bits). In contrast to RLC, as predicted by the analysis, there is a minimal value of the memory saving (for $p \simeq .9$ if $\bar{\ell} = 100$) with MCS.

Finally, note HC1 performance is extremely close to that of MCS and in particular, a minimum is observed for G(N) for the same value of p as for MCS.

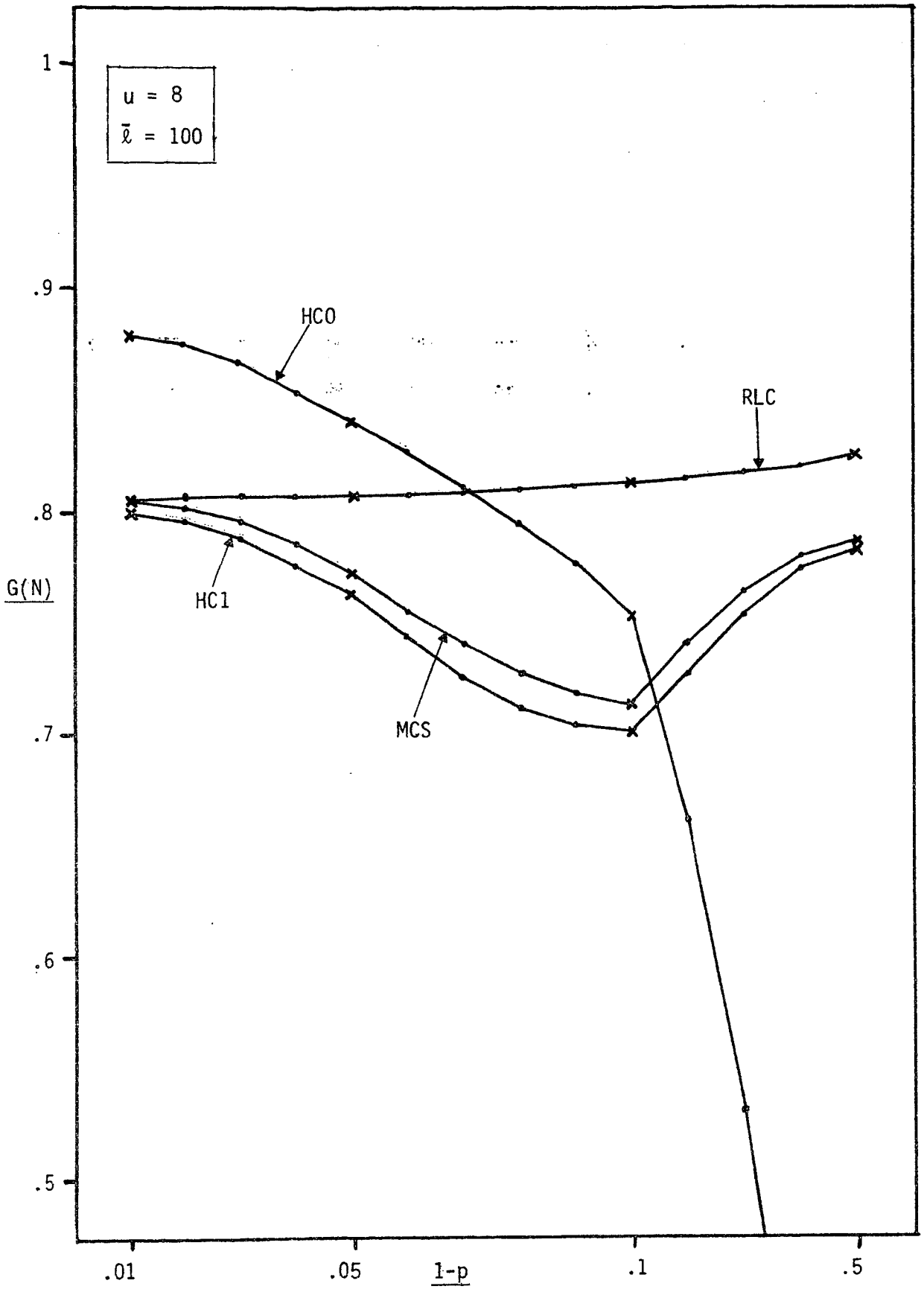

Figure 10 : HCO, HC1, RLC and MCS : memory saving versus average proportion of 1-bits (1-p) for u = 8 : SIMULATION Results

7 - CONCLUSION

In this paper, we have introduced a new compaction scheme called HCS, for cellular thematic maps decomposed into binary elementary maps. The memory saving obtained with this technique has been compared to that provided by two already existing schemes RLC [10] and MCS [3].

An analysis of the three techniques has been made, the validity of which was tested on a sample of actual data. The analysis turned out to give pessimistic predictions because of the restrictive assumptions on which it was based, especially for HCS.

A simulation experiment led to a more realistic prediction of the compaction ratio provided by these techniques, whatever the proportion of 0-bits in the maps is.

Compared to the other techniques, HCS presents the advantages of keeping set operations on maps such as intersection or union easy to be processed and of allowing variable scale (by varying the level of compaction within each map).

APPENDIX A : DERIVATION OF E(X) AND E(Y) FOR THE RLC TECHNIQUE (proof of Eqs (7) and (8)).

* $E(X) = E(X_0) + E(X_1)$
where $E(X_0)$ and $E(X_1)$ respectively represent the average length of a 0-run and a 1-run.

Let $P_{0,i}$ (resp. $P_{1,i}$) be the probability a 0-run (resp. a 1-run) is of length i. Then, we have :

$$E(X) = \sum_{i=1}^{N} i(P_{0,i} + P_{1,i})$$

As an example, if assumption A3 is verified, we have :

$$P_{0,i} = \frac{p_0^{i-1}(1-p_0)}{1-p_0^N}$$

Then, $E(X_0) = \sum_{i=1}^{N} i \frac{p_0^{i-1}(1-p_0)}{1-p_0^N}$, and when N is large, it rapidly converges to $\frac{1}{1-p_0}$.

Similarily,we have $E(X_1) = \sum_{i=1}^{N} i \frac{q_1^{i-1}(1-q_1)}{1-q_1^N}$ whose limit as N goes to infinity is equal to $\frac{1}{1-q_1}$.

Then, as N is large : $E(X) \simeq \frac{2-p_0-q_1}{(1-p_0)(1-q_1)}$ (A1)

* $E(Y) = E(Y_0) + E(Y_1)$

$= u \sum_{j=1}^{j_{max}} j(Q_{0,j} + Q_{1,j})$ where $j_{max} = \left\lceil \frac{N}{2^u-1} \right\rceil$ and $Q_{0,j}$ (resp. $Q_{1,j}$) is the probability a 0-run (resp. a 1-run) is included in the interval $](j-1)(2^u-1), j(2^u-1)]$.

If Assumption A3 is verified, we have :

$$Q_{0,j} = \left[p_0^{(j-1)(2^u-1)} - p_0^{j(2^u-1)}\right] \Big/ (1-p_0)^N$$

$$Q_{1,j} = \left[q_1^{(j-1)(2^u-1)} - q_1^{j(2^u-1)} \right] \Big/ (1-q_1)^N$$

After some algebraic manipulations and assuming N is large, we finally obtain :

$$E(Y) \simeq u \sum_{j=0}^{j_{max}-1} \left[p_0^{j(2^u-1)} + q_1^{j(2^u-1)} \right] \qquad (A2)$$

APPENDIX B : DERIVATION OF G(N) FOR THE MCS TECHNIQUE (proof of Eq. (11))

Without loss of generality, assume string C_0 starts with a 0-run. This happens with probability p_α.

Then, the average number of bits necessary to code C_0 (of length N) is :

$$S_\alpha(N) = N \left[\frac{u+2}{u} \cdot \frac{E(Y_\alpha)}{E(X_\alpha)} \right] \quad \cdots$$

where $E(Y_\alpha)$ is the average number of words (u+2 bits) necessary to code the subchain (of average length $E(X_\alpha)$ blocks of u bits) located between the occurrence of the first 0-bit of two successive 0-runs. We define similarily $E(X_i)$, $E(Y_i)$, p_i and $S_i(N)$ for $i = \beta,\gamma,\delta$.

Then, we have :

$$\frac{S(N)}{N} = \sum_{i=\alpha,\beta,\gamma,\delta} p_i \frac{S_i(N)}{N} = \frac{u+2}{u} \sum_{i=\alpha,\beta,\gamma,\delta} p_i \frac{E(Y_i)}{E(X_i)} \qquad (B1)$$

Let us first derive the $E(Y_i)$'s :

$$E(Y_\alpha) = y_\alpha + \left[E(\alpha N_\beta) y_\beta + E(\alpha N_\gamma) y_\gamma + E(\alpha N_\delta) y_\delta \right] \qquad (B2)$$

where y_i is the average number of words (u+2 bits) necessary to code a i-run, and $E(iN_j)$ the average number of "visits" to state j before coming back to state i.

We have [14] :

$$E(iN_j) = \frac{E(X_i)}{E(X_j)} \qquad (B3)$$

$$\text{Then, } E(Y_\alpha) = E(X_\alpha) \sum_{i=\alpha,\beta,\gamma,\delta} \frac{y_i}{E(X_i)} \qquad (B4)$$

From Eq. (B4), we obviously have :

$$\frac{E(Y_j)}{E(X_j)} = \sum_{i=\alpha,\beta,\gamma,\delta} \frac{y_i}{E(X_i)} \text{ , for all } j \ (j = \alpha,\beta,\gamma,\delta) \qquad (B5)$$

Plugging Eq. (B5) into Eq. (B1) and observing that $\sum_{i=\alpha,\beta,\gamma,\delta} p_i = 1$, we have :

$$\frac{S(N)}{N} = \frac{u+2}{u} \sum_{i=\alpha,\beta,\gamma,\delta} \frac{y_i}{E(X_i)} \qquad \text{(B6)}$$

Let us now calculate the $E(X_i)$'s ; $E(X_i)$ is the average number of blocks (of length u bits) in a i-subchain (starting with a i-run and ending with the last bit before the next i-run). Then we have :

$$E(X_i) = x_i + \sum_{j \neq i} x_j \, E(iN_j) \qquad \text{(B7)}$$

where x_i is the average number of blocks of a i-run.

Substituting Eq. (B3) into Eq. (B7) we finally have, as N is large :

$$\begin{aligned} E(X_\alpha) &\simeq 1/[p_0^u(1-p_0^u)] \\ E(X_\beta) &\simeq 1/[q_1^u(1-q_1^u)] \\ E(X_\gamma) &\simeq 1/[(a-p_0^u)\,(1-a+p_0^u)] \\ E(X_\delta) &\simeq 1/[(b-q_1^u)\,(1-b+q_1^u)] \end{aligned} \qquad \text{(B8)}$$

Our remaining task is the calculation of the y_i's :

First,note that $y_\gamma = x_\gamma$ and $y_\delta = x_\delta$, since to one non-monotonic block of string C_0, corresponds exactly one word in the compacted string. Since a 01-run has length k(blocks) with probability $(a-p_0^u)^{k-1}\,(1-a+p_0^u)$, as N is large ; then we have :

$$y_\gamma \simeq 1/(1-a+p_0^u) \qquad \text{(B9)}$$

and similarily $$y_\delta \simeq 1/(1-b+q_1^u) \qquad \text{(B10)}$$

One easily shows, as for the RLC technique (see Appendix 1), that

$$y_\alpha = \sum_{j=0}^{j_{max}-1} (p_0^u)^{j2^u} \qquad \text{where } j_{max} = \lceil N/2^u - 1 \rceil$$

$$y_\alpha \simeq 1/(1-p_0^{u2^u}) \quad \text{as N is large} \qquad \text{(B11)}$$

similarily, we have $y_\beta \simeq 1/(1-q_1^{u2^u})$ (B12)

Substituting Eqs (B8) to (B12) into Eq. (B6), we finally obtain Eq. (11).

REFERENCES

[1] AMIDON E.L., AKIN G.S. - "Algorithmic selection of the best method for compressing map data strings" - Comm. ACM, Vol. 14, n°12, Dec. 1971, pp. 769-774.

[2] BONFATTI F., TIBERIO P., AMBROSINI L. - "A computer-based methodology for regional intervention planning" - IEEE - SMC Int. Conf. on Cybernetics and Society, Washington, Nov. 1976.

[3] BONFATTI F., TIBERIO P. - "Data management for thematic map generation" - Computer and Graphics, Vol. 3, 1978, pp. 71-78.

[4] BOURSIER P., SCHOLL M. - "A new compaction scheme for map representation" - Int. Conf. on Image Analysis and Processing, Pavia (Italy), 22-24 Oct. 1980.

[5] BOURSIER P. - "Représentation compacte en mémoire des cartes géographiques" - Thèse de 3ème cycle, Université de Paris VI, (in preparation).

[6] BURTON W. - "Representation of many-sided polygons and polygonal lines for rapid processing"- Comm. ACM, Vol. 20, n°3, March 1977, pp. 166-171.

[7] BYROM S.T., HARDGRAVE W.T. - "Representation of sets on mass storage devices for information retrieval systems" - National Computer Conference, 1973, pp. 245-250.

[8] COWAN D.D., GRAHAM J.W., WELCH J.W. - "The efficient representation of inverted lists" - Proc. 7th S.E. Conf. on Combinatorics, Graph Theory and Computing, 1976, pp. 257-272.

[9] FREEMAN H. - "Computer processing of line-drawing images" - Computing Surveys, Vol. 6, n°1, March 1974, pp. 57-97.

[10] KORTMANN C.M. - "Redundancy reduction : a practical method of data compression" - Proc. IEEE 55, 1967, pp. 253-263.

[11] MERRILL R.D. - "Representation of contours and regions for efficient computer search" - Comm. ACM, Vol. 16, n°2, Feb. 1973, pp. 69-82.

[12] NAGY G., WAGLE S. - "Geographic data processing" - Computing Surveys, Vol. 11, n°2, June 1979, pp. 139-181.

[13] NAGY G., WAGLE S. - "Approximation of polygonal maps by cellular maps" - Comm. ACM, Vol. 22, n°9, Sept. 1979, pp. 518-525.

[14] ROSS S.M. - "Applied probability models with optimization applications" - Holden-Day, San Fransisco, 1970.

ACKNOWLEDGEMENTS : The authors thank Prof. F. BONFATTI from the University of Bologna (Italy) for having provided the data used in Section 5. They are also indebted to Dr D. POTIER and P. LEBLANC from INRIA (France) for their constructive comments and to Dr F. BANCILHON from INRIA (France) for suggesting this problem.